Proposing a New Foundation of Attack Trees in Monoidal Categories

Harley Eades III

Computer and Information Sciences, Augusta University, Augusta, GA,
heades@augusta.edu

Abstract. TODO

1 Introduction

What do propositional logic, multisets, directed acyclic graphs, source sink graphs (or parallel-series pomsets), Petri nets, and Markov processes all have in common? They are all mathematical models of attack trees [**?**], but even more than that, they can all be modeled in some form of a symmetric monoidal category[1] [**?**] – for the definition of a symmetric monoidal category see Appendix A. Taking things a little bit further, monoidal categories have a tight correspondence with linear logic through the beautiful Curry-Howard-Lambek correspondence [**?**]. This correspondence states that objects of a monoidal category correspond to the formulas of linear logic and the morphisms correspond to proofs of valid sequents of the logic. I propose that attack trees – in many different flavors – be modeled as objects in monoidal categories, and hence, as formulas of linear logic.

The Curry-Howard-Lambek correspondence is a three way relationship:

$$\begin{array}{ccccc}
\text{Categories} & \Longleftrightarrow & \text{Logic} & \Longleftrightarrow & \text{Functional Programming} \\
\text{Objects} & \Longleftrightarrow & \text{Formulas} & \Longleftrightarrow & \text{Types} \\
\text{Morphisms} & \Longleftrightarrow & \text{Proofs} & \Longleftrightarrow & \text{Programs}
\end{array}$$

By modeling attack trees in monoidal categories we obtain a sound mathematical model, a logic for reasoning about attack trees, and the means of constructing a functional programming language for defining attack trees (as types), and constructing semantically valid transformations (as programs) of attack trees. Keep in mind that as stated "semantically valid transformations as programs" is very broad, because this may turn out to be something other than morphisms between attack trees, e.g. it may turn out to be bi-simulation between processes instead.

Linear logic was first proposed by Girard [**?**] and was quickly realized to be a theory of resources. In linear logic, every hypothesis must be used exactly once.

[1] We provide a proof that the category of source sink graphs is monoidal in Appendix B.

Thus, formulas like $A \otimes A$ and A are not logically equivalent – here $\otimes$ is linear conjunction. This resource perspective of linear logic has been very fruitful in computer science. It has lead to linear logic as being a logical foundation of concurrency [?] where formulas may be considered as processes. This perspective fits modeling attack trees perfectly, because they essentially correspond to concurrent processes. Connecting attack trees to processes has been done before where they have been modeled by event-based models of concurrency like Petri nets and partially-ordered multisets (pomsets) [?]. In fact, pomsets is a model in which events (the resources) can be executed exactly once [?], and thus, has a relationship with linear logic [?]. However, connecting linear logic as a theory of attack trees is novel.

Girard's genius behind linear logic was that he isolated the structural rules – weakening and contraction – by treating them as an effect and putting them inside a comonad called the of-course exponential denoted $!A$. In fact, $!A \otimes !A$ is logically equivalent to $!A$, and thus, by staying in the comonad we become propositional. This implies that a modal of attack trees in linear logic also provides a model of attack trees in propositional logic, and a combination of the two. It is possible to have the best of both worlds.

In this short paper I introduce a newly funded research project[2] investigating founding attack trees in monoidal categories, and through the Curry-Howard-Lambek correspondence deriving a new domain-specific functional programming language called Lina for Linear Threat Analysis. I begin by defining an extension of the attack trees given in [3] in Section 2. Then I introduce a new semantics of attack trees in dialectica spaces, which is also a model of full intuitionistic linear logic, in Section 3. The final section, Section 4, discusses Lina and some of the current problems the project seeks to answer.

2 Attack Trees

In this paper I consider an extension of attack trees with sequential composition which are due to Jhawar et al. [3], but one of our ultimate goals is to extend attack trees with even more operators driven by are choice of semantics. The syntax for attack trees is defined in the following definition.

Definition 1. *The following defines the syntax of* ***Attack Trees*** *given a set of base attacks* $b \in \mathsf{B}$*:*

$$t ::= b \mid t_1 + t_2 \mid t_1 \sqcup t_2 \mid t_1 ; t_2 \mid t_1 \otimes t_2 \mid ©t$$

We denote parallel composition by $t_1 + t_2$*, choice between attacks* t_1 *and* t_2 *by* $t_1 \sqcup t_2$*, sequential composition of attacks by* $t_1; t_2$*, a new operator called orthocurrence by* $t_1 \otimes t_2$*, and finally a new operator called copy by* $©t$*.*

The following rules define the attack tree reduction relation:

[2] This material is based upon work supported by the National Science Foundation CRII CISE Research Initiation grant, "CRII:SHF: A New Foundation for Attack Trees Based on Monoidal Categories", under Grant No. 1565557.

$$\frac{}{(t_1\,\mathsf{op}\,t_2)\,\mathsf{op}\,t_3 \rightsquigarrow t_1\,\mathsf{op}(t_2\,\mathsf{op}\,t_3)}\ \textsc{Assoc} \qquad \frac{}{t_1\,\mathsf{op}_\mathsf{S}\,t_2 \rightsquigarrow t_2\,\mathsf{op}_\mathsf{S}\,t_1}\ \textsc{Sym}$$

$$\frac{}{t \sqcup t \rightsquigarrow t}\ \textsc{Choice} \qquad \frac{}{\copyright t \otimes \copyright t \rightsquigarrow \copyright t}\ \textsc{Copy}$$

$$\frac{}{(t_1 \sqcup t_2) + t \rightsquigarrow (t_1 + t) \sqcup (t_2 + t)}\ \textsc{Dist}_1 \qquad \frac{}{(t_1 \sqcup t_2); t \rightsquigarrow (t_1; t) \sqcup (t_2; t)}\ \textsc{Dist}_2$$

where $\mathsf{op} \in \{+, \otimes, ;, \sqcup\}$ *and* $\mathsf{op}_\mathsf{S} \in \{+, \otimes, \sqcup\}$*. The previous rules can be applied on any well-formed subattack tree. The equivalence relation, denoted* $\equiv$*, on attack trees is defined as the reflexive, symmetric, and transitive closure of the reduction relation* $\rightsquigarrow$*.*

The syntax given in the previous definition differs from the syntax used by Jhawar et al. [3]. First, I use infix binary operations, while they use prefix n-ary operations. However, it does not sacrifice any expressivity, because as we will see in the next section each operation is associative, and parallel composition, choice, and orthocurrence are symmetric. Thus, we can embed Jhawar et al.'s definition of attack trees into the ones defined here. The hard part of this embedding is realizing that the n-ary version of sequential composition can be modeled by the binary version, but if we have $\mathsf{SAND}(t_1, t_2, \ldots, t_n)$, then it is understood that t_1 is executed, then t_2, and so on until t_n is executed, but this is exactly the same as $t_1; t_2; \cdots; t_n$.

The second major difference is that I denote parallel composition with an operator that implies we can think of it as a disjunction, but Jhawar et al. and others in the literature seem to use an operator that implies that we can think of it as a conjunction. The semantics, however, tells us that it is really a disjunction. The parallel operation defined on source sink graphs defined by Jhawar et al. [3] can be proven to be a coproduct – see Appendix B – and coproducts categorically model disjunctions. Furthermore, parallel composition is modeled by multiset union in the multiset semantics, but we can model this as a coproduct. Lastly, the semantics I give in the next section models parallel composition as a coproduct. Thus, I claim that an operator that reflects this is for the better.

The third difference is that I denote the choice between executing attack t_1 or attack t_2, but not both, by $t_1 \sqcup t_2$ instead of using a symbol that implies that it is a disjunction. This fits very nicely with the semantics of Jhawar et al., where they collect the attacks that can be executed into a set. The semantics I given in the next section models choice directly.

The forth, and final, difference is that we extend the syntax with two new operators called orthocurrence and copy. The attack $t_1 \otimes t_2$ states that t_1 interacts with the attack t_2 in the sense that processes interact. Modeling interacting attacks allows for the more refined modeling of security critical systems. For example, one could take over a workstation on a network and funnel malicious traffic through it onto the internal network. Orthocurrence stems from process algebra, and for more examples, and a brief history of orthocurrence see [5]. In fact, this operator points out a theme to the project being described in this paper. We view attack trees as describing concurrent interacting processes. Thus, we can learn a lot from process algebra.

Orthocurrence actually hints at an interesting extension of attack trees with interacting parties. For example, it can be used to bring social engineering into the analysis where someone communicates malicious information or commands to a unsuspecting party. That is, attack trees could be extended with nodes representing people or devices communicating information from/through one process to another. I conjecture that it should be possible to extend Jhawar et al.'s model of attack trees to include orthocurrence due to the relationship their model has with partially-ordered multisets which is where orthocurrence actually originated [**?**].

The attack $\copyright t$ indicates that attack t can be copied and contracted. For example, $\copyright(t \otimes t)$ is equivalent to $\copyright t$. Thus, the attack trees given here can treat attack trees as processes/resources that cannot be freely copied and deleted, but also as propositions that can be, but even further, these two perspectives can be mixed. Semantically, $\copyright t$ is equivalent to the of-course exponential from linear logic mentioned in the introduction.

The reduction rules are a slightly extended version of equivalences given in Jhaware et al. [3] – Theorem 1. The main difference is the Copy rule which allows copies made by the copy operator to be contracted.

3 Semantics of Attack Trees in Dialectica Spaces

I now introduce a new semantics of attack trees that connects their study with two new perspectives that could highly impact future research in attack trees: intuitionistic linear logic and process calculi. Note that every construction and proof given in this section with the exception of sequential composition has been formalized in the proof assistant Agda[3]. The semantics is based on the notion of a dialectica space:

Definition 2. *A **dialectica space** is a triple (A, Q, δ) where A and Q are sets and $\delta : A \times Q \to 2$ is a relation.*

Dialectica spaces can be seen as the intuitionistic cousin [1] of Chu spaces [4]. The latter have be used extensively to study process algebra and as a model of classical linear logic, while dialectica spaces and their morphisms form a model of full intuitionistic linear logic [**?**]. I will use the intuitions often used when explaining Chu spaces to explain dialectica spaces, but it should be known that these intuitions are due to Pratt and Gupta [**?**]. However, verifying that the

[3] The complete formalization can be found at `https://github.com/heades/dialectica-spaces` which is part of a general library for working with dialectica spaces in Agda developed with Valeria de Paiva.

Chu space construction of choice and sequential composition works in dialectica spaces is novel and so is the application of this semantics to attack trees.

Intuitively, we can think of a dialectica space, (A, Q, δ), as a process where A is the set of actions the process will execute, Q is the set of states the process can enter, and for $a \in A$ and $q \in Q$, $\delta(a, q)$ indicates whether action a can be executed in state q.

Dialectica spaces form the objects of a category called $\mathsf{Dial_2(Sets)}$. This category has an abundance of structure. The definition $\mathsf{Dial_2(Sets)}$ requires the notion of a morphism between dialectica spaces.

Definition 3. *A* ***dialectica-space morphism*** *between dialectica spaces (A, Q, α) and (B, R, β) is a tuple (f, F) where $f : A \to B$ and $F : R \to Q$ such that the following weak adjointness condition holds: for any $a \in A$ and $r \in R$, if $\alpha(a, F(r))$, then $\beta(f(a), r)$.*

Proving that we have a category takes a little bit of work, but all the details can be found in the formal development. The category $\mathsf{Dial_2(Sets)}$ forms one of the earliest models of linear logic, and is the first model of intuitionistic linear logic that contains every linear operator. It is originally due to de Paiva [**?**]. For more information on how it relates to linear logic see [2].

The interpretation of attack trees into dialectica spaces, each definition is equivalent to the construction on Chu spaces [**?**], requires the construction of each operation on dialectica spaces:

Parallel Composition. Suppose $\mathcal{A} = (A, Q, \alpha)$ and $\mathcal{B} = (B, R, \beta)$ are two dialectica spaces. Then we can construct the dialectica space $\mathcal{A} + \mathcal{B} = (A + B, Q \times R, \alpha + \beta)$ where $A + B$ is the disjoint union of A and B, and $\alpha + \beta : (A + B) \times (Q \times R) \to 2$ is defined by $(\alpha + \beta)(i, x) = \alpha(i, x)$ if $i \in A$, but $(\alpha + \beta)(i, x) = \beta(i, x)$ if $i \in B$. Thus, from a process perspective we can see that $\mathcal{A} + \mathcal{B}$ executes either an action of $\mathcal{A}$ or an action of $\mathcal{B}$, but also potentially both, however this requires $\mathcal{A} + \mathcal{B}$ be a coproduct. It turns out that we can show that parallel composition is a coproduct, the details can be found in the formal development. Thus, it is associative and symmetric.

Choice. Suppose $\mathcal{A} = (A, Q, \alpha)$ and $\mathcal{B} = (B, R, \beta)$ are two dialectica spaces. Then we can construct the dialectica space $\mathcal{A} \sqcup \mathcal{B} = (A + B, Q + R, \alpha \sqcup \beta)$ where $\alpha \sqcup \beta : (A + B) \times (Q + R) \to 2$ is defined by $(\alpha \sqcup \beta)(i, j) = \alpha(i, j)$ if $i \in A$ and $j \in Q$, $(\alpha + \beta)(i, j) = \beta(i, j)$ if $i \in B$ and $j \in R$, otherwise $(\alpha + \beta)(i, j) = 0$. Thus, from a process perspective we can see that $\mathcal{A} \sqcup \mathcal{B}$ executes either an action of $\mathcal{A}$ or an action of $\mathcal{B}$, but not both. Since the actions and states of $\mathcal{A} \sqcup \mathcal{B}$ are disjoint unions it is pretty easy to show that choice forms a symmetric monoidal operator, and hence, is symmetric and associative, but it is not a coproduct, because it is not possible to define the corresponding injections.

Sequential Composition. Modeling sequential conjunction requires we extend the notion of dialectic space with the ability to determine which states of the space are initial and which are final, but this is straightforward. Suppose $\mathcal{A} = (A, Q, \alpha)$ and $\mathcal{B} = (B, R, \beta)$ are two dialectica spaces. Then we can construct the dialectica space $\mathcal{A}; \mathcal{B} = (A + B, Z, \alpha; \beta)$ where $Z = \{(q_1, q_2) \in Q \times R \mid q_1 \text{ is final in } \mathcal{A} \text{ or } q_2 \text{ is initial in } \mathcal{B}\}$, and $\alpha; \beta : (A + B) \times Z \to 2$ is defined by $(\alpha \sqcup \beta)(i, (q, r)) = \alpha(i, q)$ if $i \in A$, $(\alpha; \beta)(i, (q, r)) = \beta(i, r)$ if $i \in B$. Thus, from a process perspective we can see that $\mathcal{A}; \mathcal{B}$ will first execute the actions of $\mathcal{A}$ and then once in a final state it will begin executing actions of $\mathcal{B}$. It can be shown that sequential composition is a non-symmetric associative operation.

Orthocurrence. Suppose $\mathcal{A} = (A, Q, \alpha)$ and $\mathcal{B} = (B, R, \beta)$ are two dialectica spaces. Then we can construct the dialectica space $\mathcal{A} \otimes \mathcal{B} = (A \times B, (B \to Q) \times (A \to R), \alpha \otimes \beta)$ where $B \to Q$ and $A \to R$ denote function spaces, and $\alpha \otimes \beta : (A \times B) \times ((B \to Q) \times (A \to R)) \to 2$ is defined by $(\alpha \otimes \beta)((a, b), (f, g)) = \alpha(a, f(b)) \wedge \beta(b, g(a))$. From a process perspective the actions of $\mathcal{A} \otimes \mathcal{B}$ are actions from $\mathcal{A}$ and actions of $\mathcal{B}$, but the states are are maps (f, g) where $f : B \to Q$ and $g : A \to R$ from actions to states. This is the point of interaction between the processes $\mathcal{A}$ and $\mathcal{B}$. It is possible to show that orthocurrence is a symmetric monoidal bi-functor which is symmetric, associative, and has an identity, but for the details see the formal development.

Copying. Suppose $\mathcal{A} = (A, Q, \alpha)$ is a dialectica space. Then $©\mathcal{A} = (A, A \to Q^*, \alpha^*)$ where Q^* denotes the free monoid with carrier Q. Copying defines a functor such that if we have a morphism $f : \mathcal{B} \to \mathcal{C}$, then we obtain a morphism $©f : ©\mathcal{B} \to ©\mathcal{C}$. Furthermore, $© : \mathsf{Dial}_2(\mathsf{Sets}) \to \mathsf{Dial}_2(\mathsf{Sets})$ defines a comonad on the category of dialectica spaces, and thus, we have dialectica morphisms $\varepsilon : ©A \to A$ and $\delta : ©A \to ©©A$ satisfying the usual diagrams. Therefore, we have an isomorphism $(©\mathcal{A} \otimes ©\mathcal{A}) \cong ©\mathcal{A}$. This implies that under © we escape to propositional logic.

At this point it is straightforward to define an interpretation $[\![t]\!]$ of attack trees into $\mathsf{Dial}_2(\mathsf{Sets})$. Soundness with respect to this model would correspond to if $t_1 \rightsquigarrow t_2$, then $[\![t_1]\!] \cong [\![t_2]\!]$ where the latter takes place in $\mathsf{Dial}_2(\mathsf{Sets})$ for some suitable equivalence $\cong$ between objects. Naturally, one might choose isomorphism, but this does not hold, because the reduction rules COPY, DIST$_1$, and DIST$_2$ do not hold up to isomorphism of objects, but we conjecture if we take $\cong$ to be bi-simulation as defined for Chu spaces – see p. 63 of Gupta [**?**] – then we obtain the proper equivalences. These equivalences will need to be verified for dialectica spaces, but I do not see any barriers preventing this.

This semantics can be seen as a generalization of some existing models. Multisets, pomsets, and Petri nets can all be modeled by dialectica spaces [**?**]. However, there is a direct connection between dialectica spaces and linear logic which leads to a logical theory of attack trees.

4 Lina: A Domain Specific PL for Threat Analysis

5 Conclusion and Future Work

References

1. Valeria de Paiva. Dialectica and chu constructions: Cousins? *Theory and Applications of Categories*, 17(7):127–152, 2006.
2. Harley Eades and Valeria Paiva. *Logical Foundations of Computer Science: International Symposium, LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings*, chapter Multiple Conclusion Linear Logic: Cut Elimination and More, pages 90–105. Springer International Publishing, Cham, 2016.
3. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, SaÅ!'a RadomiroviÄ, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In Hannes Federrath and Dieter Gollmann, editors, *ICT Systems Security and Privacy Protection*, volume 455

of *IFIP Advances in Information and Communication Technology*, pages 339–353. Springer International Publishing, 2015.

4. Vaughan Pratt. Chu spaces. Notes for the School on Category Theory and Applications University of Cimbra, July 1999.
5. Vaughan R. Pratt. Orthocurrence as both interaction and observation. In *In Proc. Workshop on Spatial and Temporal Reasoning*, 2001.

A Symmetric Monoidal Categories

This appendix provides the definitions of both categories in general, and, in particular, symmetric monoidal closed categories. We begin with the definition of a category:

Definition 4. *A* ***category****, $\mathcal{C}$, consists of the following data:*

- *A set of objects $\mathcal{C}_0$, each denoted by A, B, C, etc.*
- *A set of morphisms $\mathcal{C}_1$, each denoted by f, g, h, etc.*
- *Two functions* src*, the source of a morphism, and* tar*, the target of a morphism, from morphisms to objects. If $\mathsf{src}(f) = A$ and $\mathsf{tar}(f) = B$, then we write $f : A \to B$.*
- *Given two morphisms $f : A \to B$ and $g : B \to C$, then the morphism $f; g : A \to C$, called the composition of f and g, must exist.*
- *For every object $A \in \mathcal{C}_0$, the there must exist a morphism $\mathsf{id}_A : A \to A$ called the identity morphism on A.*
- *The following axioms must hold:*
  - *(Identities) For any $f : A \to B$, $f; \mathsf{id}_B = f = \mathsf{id}_A; f$.*
  - *(Associativity) For any $f : A \to B$, $g : B \to C$, and $h : C \to D$, $(f; g); h = f; (g; h)$.*

Categories are by definition very abstract, and it is due to this that makes them so applicable. The usual example of a category is the category whose objects are all sets, and whose morphisms are set-theoretic functions. Clearly, composition and identities exist, and satisfy the axioms of a category. A second example is preordered sets, $(A, \leq)$, where the objects are elements of A and a morphism $f : a \to b$ for elements $a, b \in A$ exists iff $a \leq b$. Reflexivity yields identities, and transitivity yields composition. See the usual introductions for more examples [**?**].

Symmetric monoidal categories pair categories with a commutative monoid like structure called the tensor product. They are a categorical semantics of linear logic [**?**].

Definition 5. *A* ***symmetric monoidal category (SMC)*** *is a category, $\mathcal{M}$, with the following data:*

- *An object I of $\mathcal{M}$,*
- *A bi-functor $\otimes : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$,*

- *The following natural isomorphisms:*

$$\begin{aligned}
&\lambda_A : I \otimes A \to A \\
&\rho_A : A \otimes I \to A \\
&\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)
\end{aligned}$$

- *A symmetry natural transformation:*

$$\beta_{A,B} : A \otimes B \to B \otimes A$$

- *Subject to the following coherence diagrams:*

$$\begin{tikzcd}
((A \otimes B) \otimes C) \otimes D \arrow[r, "\alpha_{A,B,C} \otimes \mathsf{id}_D"] \arrow[d, "\alpha_{A \otimes B,C,D}"] & (A \otimes (B \otimes C)) \otimes D \arrow[dd, "\alpha_{A,B \otimes C,D}"] \\
(A \otimes B) \otimes (C \otimes D) \arrow[d, "\alpha_{A,B,C \otimes D}"] & \\
A \otimes (B \otimes (C \otimes D)) & A \otimes ((B \otimes C) \otimes D) \arrow[l, "\mathsf{id}_A \otimes \alpha_{B,C,D}"']
\end{tikzcd}$$

$$\begin{tikzcd}
(A \otimes B) \otimes C \arrow[r, "\alpha_{A,B,C}"] \arrow[d, "\beta_{A,B} \otimes \mathsf{id}_C"] & A \otimes (B \otimes C) \arrow[r, "\beta_{A,B \otimes C}"] & (B \otimes C) \otimes A \arrow[d, "\alpha_{B,C,A}"] \\
(B \otimes A) \otimes C \arrow[r, "\alpha_{B,A,C}"] & B \otimes (A \otimes C) \arrow[r, "\mathsf{id}_B \otimes \beta_{A,C}"] & B \otimes (C \otimes A)
\end{tikzcd}$$

$$\begin{tikzcd}
(A \otimes I) \otimes B \arrow[rr, "\alpha_{A,I,B}"] \arrow[dr, "\rho_A"'] & & A \otimes (I \otimes B) \arrow[dl, "\lambda_B"] \\
& A \otimes B &
\end{tikzcd}$$

$$\begin{tikzcd}
A \otimes B \arrow[d, "\beta_{A,B}"'] \arrow[dr, "\mathsf{id}_{A \otimes B}"] & \\
B \otimes A \arrow[r, "\beta_{B,A}"'] & A \otimes B
\end{tikzcd}$$

$$\begin{tikzcd}
I \otimes A \arrow[rr, "\beta_{I,A}"] \arrow[dr, "\lambda_A"'] & & A \otimes I \arrow[dl, "\rho_A"] \\
& A &
\end{tikzcd}$$

B Source Sink Graphs are Symmetric Monoidal